Parallel Algorithms for the Maximum Subarray Problem

David Tran

University of Western Ontario

CS 4402, April 2024

Table of Contents

Maximum Subarray Problem

Let A be a d-dimensional array of n reals. Find a d-dimensional subarray A' of A such that the sum of the elements of A' is maximized.

Applications

- Image processing: find the brightest region in an image

Applications

- Image processing: find the brightest region in an image
- Signal processing: find the loudest region in a sound wave

Applications

- Image processing: find the brightest region in an image
- Signal processing: find the loudest region in a sound wave
- Data mining: find the most profitable region in a dataset

Applications

- Image processing: find the brightest region in an image
- Signal processing: find the loudest region in a sound wave
- Data mining: find the most profitable region in a dataset
- Bioinformatics: find the most conserved region in a DNA sequence

History (Serial version)

- 2D version of the problem first proposed by Ulf Grenander in 1977 for maximum-likelihood estimation of patterns in images

History (Serial version)

- 2D version of the problem first proposed by Ulf Grenander in 1977 for maximum-likelihood estimation of patterns in images
- Grenander discovers $O(n^2)$ algorithm for 1D version

History (Serial version)

- 2D version of the problem first proposed by Ulf Grenander in 1977 for maximum-likelihood estimation of patterns in images
- Grenander discovers $O(n^2)$ algorithm for 1D version
- Michael Shamos discovers $O(n \log n)$ algorithm overnight

History (Serial version)

- 2D version of the problem first proposed by Ulf Grenander in 1977 for maximum-likelihood estimation of patterns in images
- Grenander discovers $O(n^2)$ algorithm for 1D version
- Michael Shamos discovers $O(n \log n)$ algorithm overnight
- Jay Kadane invents $O(n)$ in less than a minute

Table of Contents

Main Result

Theorem

There exists parallel algorithms for $d = 1$ and $d = 2$ with $O(\log n)$ time complexity. The former is optimal on an EREW-PRAM, while the latter is optimal on a CREW-PRAM.

Prefix/Suffix Sums and Maxima

$$Q = \begin{array}{|c|c|c|c|c|}\hline 1 & 2 & 3 & 4 & 5 \\ \hline\end{array}$$

Prefix/Suffix Sums and Maxima

$Q =$	1	2	3	4	5

$P(Q) =$	1	3	6	10	15

Prefix/Suffix Sums and Maxima

$$Q = \boxed{1 \mid 2 \mid 3 \mid 4 \mid 5}$$

$$P(Q) = \boxed{1 \mid 3 \mid 6 \mid 10 \mid 15}$$

$$S(Q) = \boxed{15 \mid 14 \mid 12 \mid 9 \mid 5}$$

Prefix/Suffix Sums and Maxima

$Q =$	1	2	3	4	5
$P(Q) =$	1	3	6	10	15
$S(Q) =$	15	14	12	9	5
$M_P(Q) =$	1	2	3	4	5

Prefix/Suffix Sums and Maxima

$Q =$

1	2	3	4	5

$P(Q) =$

1	3	6	10	15

$S(Q) =$

15	14	12	9	5

$M_P(Q) =$

1	2	3	4	5

$M_S(Q) =$

5	5	5	5	5

Range Query in $O(1)$

The sum of elements in the subarray $[i, j]$ is $P[j] - P[i-1]$.

$Q =$

1	2	3	4	5

$P(Q) =$

1	3	6	10	15

Sum of $Q[1, 3] = P[3] - P[0] = 10 - 1 = 9$.

Key Lemma

Lemma 1

Let Q be a 1D array of n reals. Let M_s^k be the maximum of the suffix sums of $q_0, \ldots, q_k$ and M_p^k the maximum of the prefix sums of $q_k, \ldots, q_{n-1}$. The maximum of the sums of all subarrays containing q_k is

$$Max(q_k) := M_s^k + M_p^k - q_k$$

(the sum of the largest subarrays ending at q_k and starting at q_k).

Proof (Sketch) of Lemma 1

Suppose $[A, B]$ is satisfies the above Lemma. Then any other subarray containing q_k (say $[a, b]$) has lesser sum.

	a	...	A	...	q_k	...	b	...	B	

Maximum Subarray Algorithm

1. Compute the prefix/suffix sums of Q as P, S

Maximum Subarray Algorithm

1. Compute the prefix/suffix sums of Q as P, S
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s

Maximum Subarray Algorithm

1. Compute the prefix/suffix sums of Q as P, S
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s
3. For each $1 \leq k \leq n$,

Maximum Subarray Algorithm

1. Compute the prefix/suffix sums of Q as P, S
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s
3. For each $1 \leq k \leq n$,
    1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)

Maximum Subarray Algorithm

1. Compute the prefix/suffix sums of Q as P, S
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s
3. For each $1 \leq k \leq n$,
    1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
    2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)

Maximum Subarray Algorithm

1. Compute the prefix/suffix sums of Q as P, S
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s
3. For each $1 \leq k \leq n$,
    1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
    2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
    3. $Max(q_k) = M_s^k + M_p^k - Q[k]$

Maximum Subarray Algorithm

1. Compute the prefix/suffix sums of Q as P, S
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s
3. For each $1 \leq k \leq n$,
    1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
    2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
    3. $Max(q_k) = M_s^k + M_p^k - Q[k]$
4. Return the maximum of $Max(q_k)$

Maximum Subarray Algorithm (Parallel Version)

1. Compute the prefix/suffix sums of Q as P, S (in parallel)
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s (in parallel)
3. For each $1 \leq k \leq n$, (in parallel)
   1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
   2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
   3. $Max(q_k) = M_s^k + M_p^k - Q[k]$
4. Return the maximum of $Max(q_k)$ (in parallel)

Maximum Subarray Algorithm (Analysis)

Using $O(n/\log n)$ processors on EREW-PRAM:

1. Compute the prefix/suffix sums of Q as P, S
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s
3. For each $1 \leq k \leq n$,
   1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
   2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
   3. $Max(q_k) = M_s^k + M_p^k - Q[k]$
4. Return the maximum of $Max(q_k)$

Maximum Subarray Algorithm (Analysis)

Using $O(n/\log n)$ processors on EREW-PRAM:

1. Compute the prefix/suffix sums of Q as P, S $O(\log n)$
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s
3. For each $1 \leq k \leq n$,
   1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
   2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
   3. $Max(q_k) = M_s^k + M_p^k - Q[k]$
4. Return the maximum of $Max(q_k)$

Maximum Subarray Algorithm (Analysis)

Using $O(n/\log n)$ processors on EREW-PRAM:

1. Compute the prefix/suffix sums of Q as P, S $O(\log n)$
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s $O(\log n)$
3. For each $1 \leq k \leq n$,
    1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
    2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
    3. $Max(q_k) = M_s^k + M_p^k - Q[k]$
4. Return the maximum of $Max(q_k)$

Maximum Subarray Algorithm (Analysis)

Using $O(n/\log n)$ processors on EREW-PRAM:

1. Compute the prefix/suffix sums of Q as P, S $O(\log n)$
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s $O(\log n)$
3. For each $1 \leq k \leq n$, $O(\log n)$
   1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
   2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
   3. $Max(q_k) = M_s^k + M_p^k - Q[k]$
4. Return the maximum of $Max(q_k)$

Maximum Subarray Algorithm (Analysis)

Using $O(n/\log n)$ processors on EREW-PRAM:

1. Compute the prefix/suffix sums of Q as P, S $O(\log n)$
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s $O(\log n)$
3. For each $1 \leq k \leq n$, $O(\log n)$
   1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
   2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
   3. $Max(q_k) = M_s^k + M_p^k - Q[k]$
4. Return the maximum of $Max(q_k)$ $O(\log n)$

Maximum Subarray Algorithm (Analysis)

Using $O(n/\log n)$ processors on EREW-PRAM:

1. Compute the prefix/suffix sums of Q as P, S $O(\log n)$
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s $O(\log n)$
3. For each $1 \leq k \leq n$, $O(\log n)$
    1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
    2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
    3. $Max(q_k) = M_s^k + M_p^k - Q[k]$
4. Return the maximum of $Max(q_k)$ $O(\log n)$

$T(n)$: $O(\log n)$

Maximum Subarray Algorithm (Analysis)

Using $O(n/\log n)$ processors on EREW-PRAM:

1. Compute the prefix/suffix sums of Q as P, S $O(\log n)$
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s $O(\log n)$
3. For each $1 \leq k \leq n$, $O(\log n)$
   1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
   2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
   3. $Max(q_k) = M_s^k + M_p^k - Q[k]$
4. Return the maximum of $Max(q_k)$ $O(\log n)$

$T(n)$: $O(\log n)$
$SU(n)$: $O(n)$ (Kadane)

Maximum Subarray Algorithm (Analysis)

Using $O(n/\log n)$ processors on EREW-PRAM:

1. Compute the prefix/suffix sums of Q as P, S $O(\log n)$
2. Compute the prefix maxima of S as M_p and the suffix maxima of P as M_s $O(\log n)$
3. For each $1 \leq k \leq n$, $O(\log n)$
   1. $M_s^k = M_p[k] - S[k] + Q[k]$ (range query from A to k)
   2. $M_p^k = M_s[k] - P[k] - Q[k]$ (range query from k to B)
   3. $Max(q_k) = M_s^k + M_p^k - Q[k]$
4. Return the maximum of $Max(q_k)$ $O(\log n)$

$T(n)$: $O(\log n)$
$SU(n)$: $O(n)$ (Kadane)
Efficiency: $O(n)/O(\log n \cdot n/\log n) = O(1)(!!)$

Recall: Main Result

Theorem

There exists parallel algorithms for $d = 1$ and $d = 2$ with $O(\log n)$ time complexity. The former is optimal on an EREW-PRAM, while the latter is optimal on a CREW-PRAM.

Maximum Subarray Idea: $d = 2$

2	-3	4	-1	5
-1	6	-2	7	-3
4	-2	8	-4	9

1. Pick a 2D subarray of columns.

Maximum Subarray Idea: $d = 2$

2	-3	4	-1	5
-1	6	-2	7	-3
4	-2	8	-4	9

1. Pick a 2D subarray of columns.
2. Sum across the rows to form a 1D array.

Maximum Subarray Idea: $d = 2$

2	-3	4	-1	5
-1	6	-2	7	-3
4	-2	8	-4	9

1. Pick a 2D subarray of columns.
2. Sum across the rows to form a 1D array.
3. Apply the 1D algorithm to find the maximum subarray sum.

Maximum Subarray Idea: $d = 2$

2	-3	4	-1	5
-1	6	-2	7	-3
4	-2	8	-4	9

1. Pick a 2D subarray of columns.
2. Sum across the rows to form a 1D array.
3. Apply the 1D algorithm to find the maximum subarray sum.
4. Repeat for all subarrays of columns.

Maximum Subarray Algorithm: $d = 2$

1. Replace each row by its prefix sums.

Maximum Subarray Algorithm: $d = 2$

1. Replace each row by its prefix sums.
2. Prepend a column of zeroes to the matrix.

Maximum Subarray Algorithm: $d = 2$

1. Replace each row by its prefix sums.
2. Prepend a column of zeroes to the matrix.
3. For each subarray of columns C^{gh}, $0 \leq g \leq h < n$

Maximum Subarray Algorithm: $d = 2$

1. Replace each row by its prefix sums.
2. Prepend a column of zeroes to the matrix.
3. For each subarray of columns C^{gh}, $0 \leq g \leq h < n$
    1. Collapse (sum) the rows of C^{gh} to form a a single column

Maximum Subarray Algorithm: $d = 2$

1. Replace each row by its prefix sums.
2. Prepend a column of zeroes to the matrix.
3. For each subarray of columns C^{gh}, $0 \leq g \leq h < n$
    1. Collapse (sum) the rows of C^{gh} to form a a single column
    2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}.

Maximum Subarray Algorithm: $d = 2$

1. Replace each row by its prefix sums.
2. Prepend a column of zeroes to the matrix.
3. For each subarray of columns C^{gh}, $0 \leq g \leq h < n$
    1. Collapse (sum) the rows of C^{gh} to form a a single column
    2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}.
4. Find the maximum of all M_{gh} for $0 \leq g \leq h < n$.

Maximum Subarray Algorithm: $d = 2$ (Analysis)

Assume $n^3/\log n$ processors on a CREW-PRAM, and n^3 processors on an EREW-PRAM.

1. Replace each row by its prefix sums.
2. Prepend a column of zeroes to the matrix.
3. For each subarray of columns C^{gh}, $0 \leq g \leq h < n$
    1. Collapse (sum) the rows of C^{gh} to form a 1D array.
    2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}.
4. Find the maximum of all M_{gh} for $0 \leq g \leq h < n$.

Maximum Subarray Algorithm: $d = 2$ (Analysis)

Assume $n^3/\log n$ processors on a CREW-PRAM, and n^3 processors on an EREW-PRAM.

1. Replace each row by its prefix sums. $O(\log n)$
2. Prepend a column of zeroes to the matrix.
3. For each subarray of columns C^{gh}, $0 \leq g \leq h < n$
   1. Collapse (sum) the rows of C^{gh} to form a 1D array.
   2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}.
4. Find the maximum of all M_{gh} for $0 \leq g \leq h < n$.

Maximum Subarray Algorithm: $d = 2$ (Analysis)

Assume $n^3/\log n$ processors on a CREW-PRAM, and n^3 processors on an EREW-PRAM.

1. Replace each row by its prefix sums. $O(\log n)$
2. Prepend a column of zeroes to the matrix. $O(\log n)$
3. For each subarray of columns C^{gh}, $0 \le g \le h < n$
    1. Collapse (sum) the rows of C^{gh} to form a 1D array.
    2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}.
4. Find the maximum of all M_{gh} for $0 \le g \le h < n$.

Maximum Subarray Algorithm: $d = 2$ (Analysis)

Assume $n^3/\log n$ processors on a CREW-PRAM, and n^3 processors on an EREW-PRAM.

1. Replace each row by its prefix sums. $O(\log n)$
2. Prepend a column of zeroes to the matrix. $O(\log n)$
3. For each subarray of columns C^{gh}, $0 \leq g \leq h < n$
    1. Collapse (sum) the rows of C^{gh} to form a 1D array. $O(\log n)$, using $n/\log n$ and n processors for each C^{gh} on CREW-PRAM and EREW-PRAM, respectively.
    2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}.
4. Find the maximum of all M_{gh} for $0 \leq g \leq h < n$.

Maximum Subarray Algorithm: $d = 2$ (Analysis)

Assume $n^3/\log n$ processors on a CREW-PRAM, and n^3 processors on an EREW-PRAM.

1. Replace each row by its prefix sums. $O(\log n)$
2. Prepend a column of zeroes to the matrix. $O(\log n)$
3. For each subarray of columns C^{gh}, $0 \leq g \leq h < n$
   1. Collapse (sum) the rows of C^{gh} to form a 1D array. $O(\log n)$, using $n/\log n$ and n processors for each C^{gh} on CREW-PRAM and EREW-PRAM, respectively.
   2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}. $O(\log n)$
4. Find the maximum of all M_{gh} for $0 \leq g \leq h < n$.

Maximum Subarray Algorithm: $d = 2$ (Analysis)

Assume $n^3/\log n$ processors on a CREW-PRAM, and n^3 processors on an EREW-PRAM.

1. Replace each row by its prefix sums. $O(\log n)$
2. Prepend a column of zeroes to the matrix. $O(\log n)$
3. For each subarray of columns C^{gh}, $0 \leq g \leq h < n$
    1. Collapse (sum) the rows of C^{gh} to form a 1D array. $O(\log n)$, using $n/\log n$ and n processors for each C^{gh} on CREW-PRAM and EREW-PRAM, respectively.
    2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}. $O(\log n)$
4. Find the maximum of all M_{gh} for $0 \leq g \leq h < n$. $O(\log n)$

Maximum Subarray Algorithm: $d = 2$ (Analysis)

Assume $n^3/\log n$ processors on a CREW-PRAM, and n^3 processors on an EREW-PRAM.

1. Replace each row by its prefix sums. $O(\log n)$
2. Prepend a column of zeroes to the matrix. $O(\log n)$
3. For each subarray of columns C^{gh}, $0 \le g \le h < n$
   1. Collapse (sum) the rows of C^{gh} to form a 1D array. $O(\log n)$, using $n/\log n$ and n processors for each C^{gh} on CREW-PRAM and EREW-PRAM, respectively.
   2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}. $O(\log n)$
4. Find the maximum of all M_{gh} for $0 \le g \le h < n$. $O(\log n)$

$T(n)$: $O(\log n)$

Maximum Subarray Algorithm: $d = 2$ (Analysis)

Assume $n^3/\log n$ processors on a CREW-PRAM, and n^3 processors on an EREW-PRAM.

1. Replace each row by its prefix sums. $O(\log n)$
2. Prepend a column of zeroes to the matrix. $O(\log n)$
3. For each subarray of columns C^{gh}, $0 \leq g \leq h < n$
    1. Collapse (sum) the rows of C^{gh} to form a 1D array. $O(\log n)$, using $n/\log n$ and n processors for each C^{gh} on CREW-PRAM and EREW-PRAM, respectively.
    2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}. $O(\log n)$
4. Find the maximum of all M_{gh} for $0 \leq g \leq h < n$. $O(\log n)$

$T(n)$: $O(\log n)$
$SU(n)$: $O(n^3)$

Maximum Subarray Algorithm: $d = 2$ (Analysis)

Assume $n^3/\log n$ processors on a CREW-PRAM, and n^3 processors on an EREW-PRAM.

1. Replace each row by its prefix sums. $O(\log n)$
2. Prepend a column of zeroes to the matrix. $O(\log n)$
3. For each subarray of columns C^{gh}, $0 \le g \le h < n$
    1. Collapse (sum) the rows of C^{gh} to form a 1D array. $O(\log n)$, using $n/\log n$ and n processors for each C^{gh} on CREW-PRAM and EREW-PRAM, respectively.
    2. Apply the 1D algorithm to find the maximum subarray sum M_{gh} for C^{gh}. $O(\log n)$
4. Find the maximum of all M_{gh} for $0 \le g \le h < n$. $O(\log n)$

$T(n)$: $O(\log n)$
$SU(n)$: $O(n^3)$
Efficiency: $O(n^3)/O(\log n \cdot n^3/\log n) = O(1)$

Maximum Subarray Algorithm: $d > 2$

In general, for each of the $\binom{n}{2}^{d-2}$ d-dimensional subarrays, we can collapse them to a $d-1$-dimensional subarray and apply the $d-1$ algorithm to it in $O(\log n)$ time.

Remark

The d-dimensional maximum subarray problem can be solved in $O(\log n)$ time with $n \cdot \binom{n}{2}^{d-1}$ processors on a CREW-PRAM, and $n^2 \cdot \binom{n}{2}^{d-1}$ processors on an EREW-PRAM.

Table of Contents

Recent Work

- (1998) A (very slightly) sub-cubic algorithm for the 2D serial version using matmul
- (2004) 2D parallel version of the algorithm designed for BSP/CGM models
- (2017) 2D parallel version of the algorithm (among other problems) with optimal communication complexity on the systolic array model (FGPA, ASIC)

Citations

- Perumalla, K., & Deo, N. (1995). *Parallel algorithms for maximum subsequence and maximum subarray*. Parallel Processing Letters. World Scientific Publishing Company.
- Alves, C.E.R., Cáceres, E.N., Song, S.W. (2004). BSP/CGM Algorithms for Maximum Subsequence and Maximum Subarray.
- Bae, S.E.; Shinn, T.-W.; Takaoka, T. Efficient Algorithms for the Maximum Sum Problems. Algorithms (2017), 10, 5. https://doi.org/10.3390/a10010005